

ГЛАВА XX.

Постановка основных задач

*Задача – это попасть в цель,
а проблема – когда целятся в тебя.*

*Наши цели ясны, задачи определены!
За работу, товарищи!*

Н.С. Хрущёв

Обучение по размеченным данным

Рассмотрим постановки основных задач машинного обучения. Начнём с **обучения с учителем (Supervised Learning)**, которое также более удачно¹ называют **обучением с размеченными данными / метками**. В этой задаче доступные данные представляются в следующем виде:

$$X_{\text{train}} = \{(x_1, y_1), \dots, (x_m, y_m)\}.$$

Постулируется, что есть некоторая зависимость $y: X \rightarrow Y$, которую часто называют **целевой функцией**² (**target**), а $y(x)$ – **меткой (label)** объекта x . Зависимость нам задана лишь на конечном наборе точек (это задание как раз и отображается в исходных данных):

$$\begin{aligned} y(x_1) &= y_1, \\ &\dots \\ y(x_m) &= y_m, \end{aligned}$$

x_i – **объект**³ (наблюдение, **observation, example, instance, object**). X – **пространство объектов (входов)**, Y – **пространство меток / значений целевого признака (выхода)**. В обозначении исходных данных использован индекс «train» – мы также будем называть это множество

Часто различают понятие «объект» и «описание объекта», например признаковое.

¹ Термин «учитель» (teacher) уже занят в глубоком машинном обучении для обозначения нейронной сети, которую используют для обучения другой нейронной сети: ученика (student).

² Также y могут называть **целевой переменной / значением (target/response/outputs dependent variable)**.

³ Иногда более формально говорят «**описание объекта**», т.к. объект это например пациент в клинике, а его описание – перечень результатов анализов (в машинном обучении используется второе, а не физический пациент).

обучающей выборкой (train set), хотя дальше будет ясно, что данные обычно делят на подмножества и лишь одно из них является обучающим для алгоритмов машинного обучения. В машинном обучении с размеченными данными обычно следующие проблемы центральные:

1. **Восстановление целевой зависимости** (научиться восстанавливать метки $y(x)$ новых объектов x , т.е. найти зависимость целевого значения от описания объекта).

Для математиков задача обучения по меткам – задача экстраполяции!

2. **Интерпретация** (после восстановления объяснить, как устроена функция $y(x)$).

3. **Оценка качества полученного решения** (например, оценка того, как часто ошибаемся при предсказании $y(x)$, насколько ошибаемся и т.п.).

Есть следующие типы задач обучения с учителем:

- **классификация** (когда множество меток Y конечное и, как правило, небольшое: $|Y| = l \ll \infty$),
- **регрессия**¹ (когда метки – вещественные числа: $Y = \mathbb{R}$).

Классификация может быть на l **непересекающихся классов (multiclass classification)**, тогда $Y = \{1, 2, \dots, l\}$ или на l **пересекающихся классов (multi-label classification)**, тогда удобнее считать, что $Y = \{0, 1\}^l$ (так обозначается l -я декартова степень множества, т.е. множество Y состоит из бинарных l -мерных векторов). В последнем случае каждому объекту будет сопоставлен l -мерный бинарный вектор, будем говорить, что объект принадлежит j -му классу, если j -я координата этого вектора равна 1 (меткой тут называют и сам вектор, но чаще элементы множества $\{1, 2, \dots, l\}$, из которого несколько может быть сопоставлено объекту, отсюда термин «multi-label»). Например, вектор классификации

$$y(x) = (1, 0, 1, 0, 0)$$

обозначает, что в задаче классификации с пятью классами объект x принадлежит только первому и третьему.

¹ После знакомства с типами признаков можно будет сказать, что в задаче классификации целевой признак категориальный, а в задаче регрессии – вещественный.

Отдельно выделяют задачу **бинарной классификации** – случай двух классов, в котором чаще используют такие метки классов:

$$Y = \{0, 1\} \text{ или } Y = \{-1, +1\}$$

(выбор кодирования меток, как мы увидим дальше, часто обусловлен удобством его использования, например, для описания алгоритма машинного обучения решения задачи). При $Y = \mathbb{R}^l$ говорят о **многомерной задаче регрессии**.

Иногда будем рассматривать **скоринговые постановки**¹ задач машинного обучения с размеченными данными, в которых необходимо получить не метки объектов, а распределения на множестве меток. Например, в задаче бинарной классификации с $Y = \{0, 1\}$ алгоритм должен на каждом объекте x получать значение $a(x) \in [0, 1]$ из отрезка, которое можно будет интерпретировать как уверенность (как вероятность – если удастся доказать обоснованность такой интерпретации) принадлежности объекта к классу 1. В задаче с l классами определяется l степеней уверенности:

$$a(x) = (\alpha_1, \dots, \alpha_l) \in [0, 1]^l,$$

для непересекающихся классов логично потребовать $\alpha_1 + \dots + \alpha_l = 1$.

Приведём **примеры задач обучения с метками**. Задача 1 – **классификация спама**, есть почтовый ящик, в который приходят письма, часть из них пользователю не интересна, и он помещает их в папку «спам», другая часть интересна (он их полностью читает, отвечает на них и т.п.). Хочется, чтобы процедуру фильтрации писем делал некоторый алгоритм. В этой задаче множество объектов X – множество электронных писем, множество меток $Y = \{\text{спам}, \text{норма}\}$, обучающую выборку создаёт пользователь с помощью сортировки писем². Можно считать, что всем письмам папки «спам» приписана соответствующая метка, остальным – метка «норма». Получили бинарную задачу классификации. Если бы меток было больше (по названию папок), то получили бы задачу классификации на непересекающиеся классы.

Задача 2 – **медицинская диагностика**. Пациентам в клинике на основе собранных данных (данные анкеты, медицинская книжка, история болезни,

¹ Этот термин не сильно распространён.

² Можно также использовать доступные данные в интернете, например, коллекции спамовых писем.

результаты анализов и т.д.) делается диагностика¹ (упрощённо: болен или нет определённой болезнью). Желательно создать «электронного помощника», который, учитывая накопленную статистику, подсказывает врачу диагноз. В этой задаче множество объектов X – множество пациентов (точнее, их описаний), множество меток Y в простейшем случае двухэлементное («болен», «здоров»). В более сложной требуется определить вероятность каждой болезни из некоторого конечного фиксированного набора: $Y = [0, 1]^l$. Обучающая выборка формируется из накопленной статистики (какой-то пациент оказался больным, какой-то здоровым и т.д.).

Задача 3 – прогнозирование цен акций: в каждый момент времени предсказывать цену акции (набора акций) через некоторый фиксированный интервал времени. В такой постановке объект – ситуация на рынке (текущие котировки, метаинформация о компаниях, торгах, содержание твитов и т.п.), метка – цена на акцию через интервал времени или несколько цен (тогда $Y = \mathbb{R}^l$). Обучающая выборка формируется «из прошлого»: в каких ситуациях как себя вели цены.

Сделаем ещё несколько замечаний по терминам. **Задачами прогнозирования** называют задачи, в которых объекты можно упорядочить по времени и нас интересуют метки «более свежих» объектов, чем были в обучающей выборке. Говорят, что алгоритм обучается на данных из прошлого (до какого-то фиксированного момента времени), а работает на данных из будущего (после этого момента). В примере выше задача прогнозирования цен акций имеет смысл именно как задача прогнозирования, а фильтрация спама имеет общий смысл (можно автоматически отфильтровать все письма, которые накопились в почтовом ящике, в том числе и довольно старые).

Описание объектов

Пространство объектов может быть практически любым, в реальных задачах объектами (точнее их описаниями) являются:

Объекты и метки
могут быть любыми!

- медицинские истории, результаты анализов, экспертные мнения,
- финансовые проекты / экономические ситуации,

¹ Конечно, тут неявно учитывается и сам факт того, что человек оказался пациентом (есть подозрение заболевания). Также при диагностике помогает интуиция и опыт врача.

- месторождения, объекты недвижимости,
- тексты (как отдельные сообщения в мессенджерах и социальных сетях, так и целые книги),
- сигналы / временные ряды / последовательности,
- звуки, изображения, видеоролики,
- векторы / множества / графы.

В данной книге, в основном, тщательно рассматривается случай, когда $X = \mathbb{R}^n$ — n -мерное **признаковое пространство**, т.е. объект задаётся n -мерным вещественным вектором:

$$x_i = (x_{i1}, \dots, x_{in})$$

— **признаковым описанием**, элементы вектора называются **признаками** (inputs, attributes, repressors, properties, covariates, features, variables). Вообще, в задаче в признаковой постановке данные удобно представить в виде **матрицы «объект-признак»** (матрица данных, data matrix), см. табл. XX.1. По строкам в такой матрице записаны признаковые описания объектов, а по столбцам — значения конкретных признаков. Иногда отдельный столбец помечают как **целевой** (в нём содержатся метки, которые надо восстанавливать по значениям в других столбцах), но чаще в матрице данных указываются доступные для наблюдения данные, а (вообще говоря неизвестные) метки указываются отдельно¹.

плохой_клиент	линии	возраст	поведение_30-59_дней	Debt_Ratio	доход	число_кредитов
0	0.111673	46	0	1.329588	800.0	8
0	0.044097	69	0	0.535122	3800.0	10
0	0.047598	77	0	0.169610	3000.0	7
0	0.761149	58	1	2217.000000	NaN	4
0	0.690684	55	0	0.432552	12416.0	7

Табл. XX.1. Матрица данных.

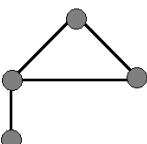
Для обозначения j -го признака объекта x иногда будет удобно использовать обозначение $f_j(x)$. **Извлечением признаков** называется функция

$$f_{1:n}(x) = (f_1(x), \dots, f_n(x)) : X \rightarrow \mathbb{R}^n$$

¹ Часто целевых значений изначально нет, их надо придумать. Например, ниже рассмотрим проблему оттока клиентов, изначально даны описания клиентов. В зависимости от выбора целевого признака меняется задача.

(ну или процесс применения этой функции на практике), т.е. которая из описания объекта (произвольной природы) получает вектор признаков. На практике от извлечённых признаков существенно зависит качество решения задачи машинного обучения: чем лучше признаки, тем более простые алгоритмы нужны для решения. Приведём ниже несколько примеров извлечения признаков для таких объектов как строка, адрес электронной почты и граф. Скорее всего, они будут понятны без дополнительных комментариев.

Чем лучше признаки, тем проще алгоритмы восстановления меток.

объект	признаки
строка «ABDDBAABB»	длина = 9 число вхождений A = 3 число вхождений B = 4 число вхождений C = 0 число вхождений D = 2 число уникальных символов = 3
почта «dag@pk.tsu.ru»	длина (до @) = 3 доменов = 3 $I[1 \text{ уровень} = \text{ru}]^1 = 1$ $I[1 \text{ уровень} = \text{com}] = 0$
граф 	число вершин = 4 число рёбер = 4 число компонент связности = 1 максимальная степень вершины = 3

Визуализация задач

Простейшие задачи машинного обучения можно «изобразить на картинке». Ниже показано несколько модельных задач, подобные иллюстрации будем использовать и дальше. На рис. XX.1 – задача регрессии с одним вещественным признаком x^2 и одним целевым признаком y , тоже вещественным, в этом случае объект с меткой задаётся точкой на плоскости с координатами (x, y) . Решать задачу можно поиском кривой, которая проходит через все точки или

¹ Используем $I[\text{условие}]$ для обозначения индикатора: функции, которая принимает значение 1 при выполнении условия, иначе – 0.

² Если быть формалистами, то $x = (f_1(x))$ – одноэлементный вектор, но мы будем в данном случае отождествлять этот вектор с его единственным элементом.

рядом с ними (значения признаков могут быть известны с ошибками, поэтому нет смысла восстанавливать зависимость «как есть»). На рис. XX.1 пунктиром показана истинная зависимость, метки известны с точностью до некоторого шума, поэтому точки обучающей выборки не лежат на пунктире, построены полиномы разных степеней (1, 3 и 7).

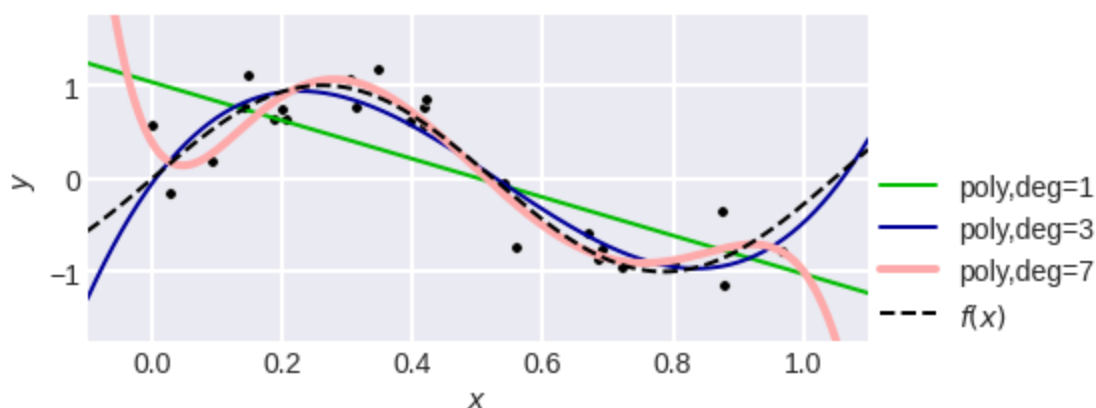


Рис. XX.1. Иллюстрация задачи регрессии.

Заметим, что на рис. XX.1 истинную зависимость мы обозначили через $f(\cdot)$, чтобы отличать её от функции $y(\cdot)$, которую логичнее назвать **функцией разметки**, поскольку на каждом i -м объекте целевое значение нам известно с точностью до шума $\varepsilon_i \in \mathbb{R}$:

$$y_i \equiv y(x_i) = f(x_i) + \varepsilon_i.$$

На рис. XX.2 показана иллюстрация задачи бинарной классификации с двумя вещественными признаками. Каждый объект представляется точкой с координатами (признак 1, признак 2) и раскрашивается цветом, соответствующим своему классу. Алгоритм бинарной классификации будет для каждого объекта формировать одну из двух возможных меток, а с геометрической точки зрения, делить плоскость на две части **разделяющей поверхностью (decision boundary)**, она показана чёрной линией): «объекты первого класса по мнению алгоритма» и «объекты второго класса по мнению алгоритма». Большинство алгоритмов классификации могут оценивать вероятность принадлежности к классам, на рис. XX.2 (слева) вероятность принадлежности к классу 1 показана оттенком серого (чем темнее, тем больше вероятность). На рис. XX.2 (справа) области, на которые разделяющая поверхность делит пространство (в данном случае, плоскость), раскрашены разным цветом, чтобы были понятны ответы классификатора в каждой точке пространства.

Заметим, что сложно наглядно изобразить задачи с большим числом признаков.

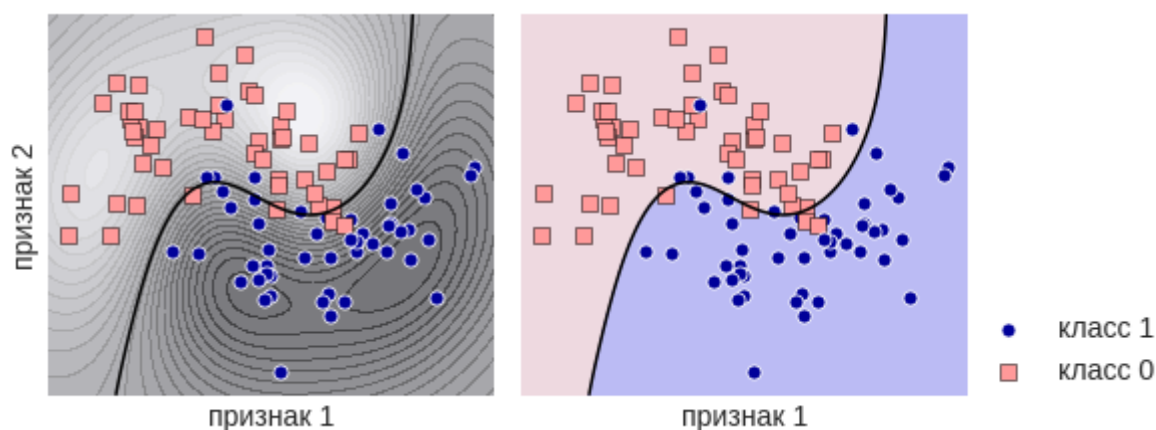


Рис. XX.2. Иллюстрация задачи классификации: ответ с вероятностью принадлежности классу 1 (слева – оттенком серого), ответ с разделением на классы (справа – разными цветами).

Решение задачи – алгоритм

Как было сказано, в задаче обучения с метками надо для каждого объекта $x \in X$ уметь достаточно точно «предсказывать» метку $y \in Y$. На практике для такого предсказания строится алгоритм. Здесь и далее под **алгоритмом** в данном контексте мы понимаем функцию

$$a(x): X \rightarrow Y,$$

которую можно эффективно реализовать в виде программы, и которая

- допускает вычисление за приемлемое время,
- использует ограниченный набор ресурсов,
- учитывает специфику, связанную с вычислениями на компьютере (впрочем, на это мы не всегда будем обращать внимание).

На практике требований к алгоритму (или модели в целом) гораздо больше¹:

- **Качество** (Predictive Accuracy) – формализует точность предсказания меток (см. выше),

¹ Мы это рассмотрим в главе **ML System Design**. Набор требований, который предъявляется к алгоритму, зависит от постановки задачи.

- **Эффективность** (Efficiency) – время обучения и использования,
- **Робастность** (Robustness) – устойчивость к некачественным данным (шуму, пропускам в данных и т.п.),
- **Масштабируемость** (Scalability) – возможность использования при увеличении объёма данных,
- **Интерпретируемость** (Interpretability) – возможность объяснения результатов модели,
- **Компактность** (Compactness) – небольшие затраты на хранение модели.

Отметим, что кроме разной степени выполнения описанных свойств, модели алгоритмов машинного обучения различаются функциональной выразимостью (какие задачи они могут приемлемо решать) и геометрией разделяющих поверхностей. На рис. XX.3 представлены решения одной конкретной задачи разными алгоритмами, точнее их разделяющие поверхности. Все эти алгоритмы являются представителями разных моделей, каждой модели будет посвящена отдельная глава книги (и неплохо бы после их изучения по рисунку понимать, решение алгоритма какой модели показано).

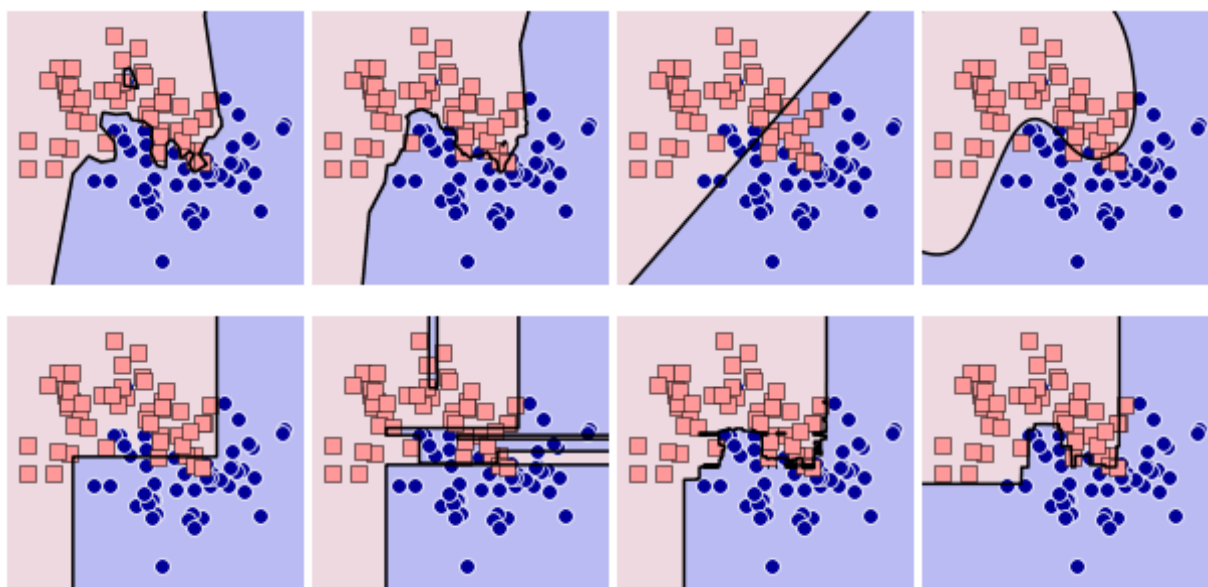


Рис. XX.3. Разделяющие поверхности разных бинарных классификаторов.

Алгоритмы для решения задач классификации называются **классификаторами**, а для решения задач регрессии – **регрессорами**. Часто алгоритмы выбираются исходя из экспертных знаний о предметной области. В **параметрическом подходе** задаётся аналитический вид модели с явной

зависимостью от параметров, например, **линейная модель** выглядит так:
 $a(x; w, w_0) = w^T x + w_0$ для n -мерного вектора параметров $w \in \mathbb{R}^n$ или

$$a(X_1, \dots, X_n) = w_0 + w_1 X_1 + \dots + w_n X_n,$$

w_i – веса (weights) / параметры (parameters) модели, параметр w_0 называют смещением (bias). Параметры оцениваются с помощью **подгонки на данных обучения** (fitting the model to training data), в идеале на объектах обучающей выборки должны выполняться равенства

$$a(x_i) = y_i, \quad i = 1, 2, \dots, m$$

(далее увидим, что точное выполнение равенств для линейной модели не всегда возможно, а для нелинейных может быть нежелательно). Если есть гипотеза, что целевые значения достаточно точно можно линейно выразить через значения признаков, то разумно использовать линейную модель. Но такая модель может оказаться слишком простой для решаемой задачи, поэтому её можно усложнить, часто используют полиномиальную модель второй степени:

$$a(X_1, \dots, X_n) = w_0 + \sum_t w_t X_t + \dots + \sum_{i,j} w_{ij} X_i X_j.$$

Восстановление целевой зависимости

Для пояснения понятия «восстановление целевой зависимости» введём **функцию ошибки (error / loss function)** $L(y, a)$ ¹, которая формализует ошибку на объекте x :

$$L(y(x), a(x)) \in \mathbb{R}^+,$$

где $a(x)$ – ответ нашего алгоритма a на объекте x ,
 $y(x)$ – метка этого объекта. Например,
представляется разумным использовать такие функции:

Иногда алгоритм называют также «гипотезой».

в задаче регрессии – $L(y, a) = |y - a|$,

в задаче классификации – $L(y, a) = I[y \neq a]$.

¹ Здесь есть некоторая коллизия: y означает и искомую зависимость и аргумент функции, но это не приводит к путанице.

Более подробно мы будем изучать такие функции в нескольких главах, т.к. понимание, как оценивается решение, важно для его получения.

Если объекты имеют вероятностную природу, т.е. есть некоторая вероятностная мера $P(x, y)$, описывающая распределение пар («объект», «метка»), то логично решать такую задачу оптимизации

$$\int_{X \times Y} L(y, a(x)) dP(x, y) \rightarrow \min, \quad (\text{XX.1})$$

минимизировать т.н. **теоретический риск**. Минимизация проводится по алгоритмам a в некотором семействе (пространстве) алгоритмов, далее такие семейства мы будем называть моделями. Более формально, **модель** – параметрическое семейство алгоритмов $A = \{a(x; w)\}_{w \in W}$, пример:

$$A = \{a(x; w) = w^T x : \mathbb{R}^n \rightarrow \mathbb{R}\}_{w \in \mathbb{R}^n} -$$

– модель линейных однородных функций от признаков, здесь каждый алгоритм определяется значениями вектора параметров $w \in \mathbb{R}^n$.

На практике мера в формуле (XX.1) не известна, и можно лишь оценить теоретический риск с помощью «**эмпирического риска**»: ошибки на обучающей выборке

$$L(a, X_{\text{train}}) = \frac{1}{m} \sum_{i=1}^m L(y(x_i), a(x_i)), \quad (\text{XX.2})$$

где $X_{\text{train}} = \{(x_1, y_1), \dots, (x_m, y_m)\}$ – **обучающая выборка** (или по-простому «**обучение**» – не путать с процессом построения алгоритма, см. ниже). Здесь мы усреднили ошибки на всех объектах обучения, это один из вариантов определения ошибки на всей выборке, но далеко не единственный (и не всегда самый лучший). Тогда искомый алгоритм

$$a^* = \arg \min L(a, X_{\text{train}}),$$

достаточно определить параметры w алгоритма $a(x; w)$, при которых достигается минимум эмпирического риска. Таким образом, **обучение** (теперь уже имеется в виду процесс) – это определение параметров алгоритма, как правило, производится с помощью оптимизации значения функции ошибки или её модификации (далее об этом поговорим) на обучающей выборке.

По-простому, обучение – интеллектуальный перебор параметров (алгоритмов).

На самом деле, интересна не ошибка на обучении (training error), а ошибка на тех данных, с которыми алгоритм будет иметь дело в будущем. И здесь возникает важное понятие – **обобщающая способность** (generalization). Можно его формализовать через сравнение значений

$$L(a, X_{\text{train}}) \vee L(a, X_{\text{test}}),$$

т.е. ошибок на обучающей X_{train} и тестовой X_{test} выборках (для нас сейчас тестовая выборка – это размеченная подвыборка исходных данных, которую отложили и не использовали при обучении). Последнюю ошибку называют **тестовой** (Test Error) или **ошибкой обобщения** (Generalization Error), более строго часто говорят о матожидании ошибки на новых данных. Если при небольшой ошибке на обучении у алгоритма небольшая тестовая ошибка, то считается, что у него хорошая обобщающая способность: зависимости, которые он обнаруживает в обучающей выборке «обобщаются» на другие данные. Легко построить алгоритм, который даёт минимальную ошибку на обучении. Например, можно просто запомнить обучающую выборку, и для каждого объекта, для которого надо восстановить метку, осуществлять поиск – был ли он в обучении. Если был, то выдавать его метку из обучения, если нет – случайную метку или отказываться от классификации:

обучение \neq запоминание

$$a(x) = \begin{cases} y_i, & \exists i \in \{1, 2, \dots, m\}: x = x_i, \\ \Delta, & \text{иначе,} \end{cases}$$

здесь Δ – отказ от классификации. Такой алгоритм не способен «разумно» восстанавливать метки неизвестных объектов и про него говорят, что у него **нет никакой обобщающей способности**. Позже подробно изучим понятия, имеющие отношение к обобщающей способности: недообучение, переобучение, сложность, а также её верификации: отложенная выборка, контроль и т.п. Пока заметим, что чем сложнее модель, тем проще настроиться на данные (получить низкую ошибку на обучении), но также проще и **переобучиться (overfit)**: получить существенно большую ошибку на тестовой выборке. Из примеров выше, полиномиальная модель степени 2 сложнее линейной, которая к тому же ещё и хорошо интерпретируемая:

Простые модели, как правило, надёжны (оценка их ошибки соответствует действительности).

- легко объяснить, как она работает,

- легко объяснить, почему получен именно такой ответ в каждом конкретном случае.

Почему машинное обучение является отдельной дисциплиной

Основная задача построения алгоритма машинного обучения была сведена к задаче минимизации эмпирического риска (XX.2), почему тогда машинное обучение не является разделом теории оптимизации? Причины следующие:

1. Поскольку не известна мера в теоретическом риске (XX.1), который надо бы минимизировать, то решается «неправильная» задача оптимизации эмпирического риска (XX.2). Выше показано, что формальное её решение из-за эффекта запоминания может приводить к нежелательным последствиям. Кроме того, функционал для минимизации (XX.2) часто модифицируют. Правильный выбор модификации (регуляризация, учёт контекста – проблемно-ориентированное моделирование и т.п.) – это особое умение, которое и изучается в машинном обучении.

2. Оптимизация производится не в классе функций, а в классе алгоритмов: функций с дополнительными требованиями. Среди них есть и «относительно простые» функции, но также и «довольно сложные», которые не рассматриваются в классических курсах по оптимизации (реализуются случайными лесами, нейронными сетями и т.п.).

3. При решении задачи машинного обучения есть контекст: мы знаем смысл задачи и зачем её решать, это порождает специальные приёмы, которые не применяются при классической оптимизации, например, аугментацию (когда алгоритм учит определять, кто изображён на картинках, картинки немного модифицируют, при этом модификации не меняют семантики изображений: зашумляют, поворачивают и т.п.).

Решается прикладная задача, а не конкретная задача оптимизации.

Схема решения задачи машинного обучения

Опишем типичный процесс решения задачи машинного обучения «с нуля» (т.е. с понимания проблемы и постановки задачи) и «до получения результатов» (т.е. выполнения проекта с машинным обучением). Более подробно разберём её в **главе «ML System Design»**.

1. Уточнение и постановка задачи (Problem Definition)

- понимание бизнес-проблемы,
- математическая формализация (постановка задачи машинного обучения).

Здесь происходит постановка задачи машинного обучения, определяется ДНК (дано-найти-критерий). Заметим, что формально одна и та же проблема порождает разные задачи машинного обучения. Пример для проблемы ухода клиентов (отказа от услуг компании): можно предсказывать уходящих и предлагать им бонусы и скидки с целью задержать, можно понять причину ухода и попытаться её устранить, например, предсказывать поломки оборудования из-за которых клиенты недовольны услугами. Не все задачи можно решить эффективно, не все – методами машинного обучения, не для всех есть данные в нужном объёме и надлежащего качества.

2. Сбор, подготовка и анализ данных (Data Mining)

- понимание исходных данных,
- сбор данных (Data collection),
- предобработка данных (Data cleaning) / подготовка данных для модели,
- разведочный анализ (Exploratory Data Analysis).

На этом этапе мы убеждаемся, что данные есть (и будут при функционировании нашего алгоритма), они позволяют решить задачу (хотя точный прогноз качества решения может быть затруднителен), подготавливаем их для моделирования.

3. Выбор

- алгоритма:
 - модели алгоритмов (Model selection),
 - способа обучения (гиперпараметры, методы оптимизации),
- контроля:
 - функции ошибки (Metric selection),
 - способа контроля (разбиение train/test/valid),
- признаков:

- генерация признаков (Data coding, feature engineering)
- селекция признаков (feature selection)

На этом этапе определяется, как будет решаться задача. Не исключено, что описанные выборы делаются перебором, например перебираются разные модели алгоритмов (последовательно применяя и следующие этапы).

4. Обучение (Fitting)

Как раз на этом этапе и выполняется машинное обучение – подбор параметров нашего алгоритма.

5. Оценка алгоритма (Evaluation)

- **предсказание (Prediction)** – получение ответов на новых данных,
- (возможно) **пост-обработка (Post-processing)**,
- **проверка качества.**

Мы получаем результаты работы алгоритма на новых данных (которые не использовались при обучении) и оцениваем качество его работы. Проверке качества будет посвящена **отдельная глава**, пока для понимания, как можно проверять алгоритмы, достаточно иллюстрации на рис. XX.4. Все данные делятся на две части: обучающая и тестовая выборки (train / test). На обучающей мы подбираем параметры алгоритма (он схематично показан «чёрным ящиком») – производим обучение, на тестовой получаем метки по описаниям объектов и сравниваем их с истинными.

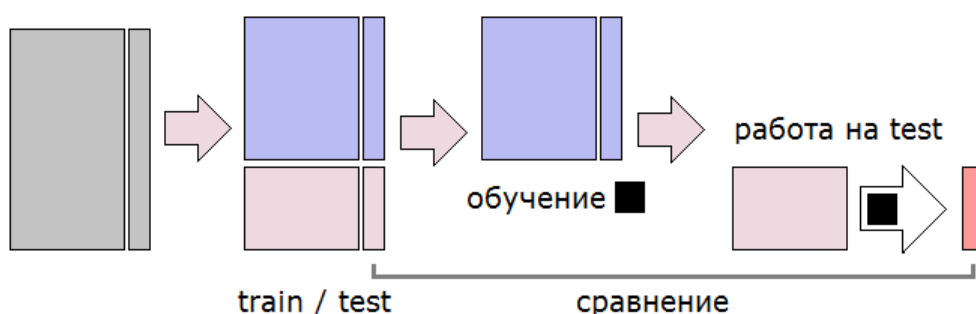


Рис. XX.4. Простейшая схема проверки качества алгоритма.

6. Использование (Deploy / Release), онлайн-оценивание (Online evaluation), отладка (Debug) и поддержка (Maintenance).

Если алгоритм нас устраивает, то его «выкладывают в прод» и он начинает приносить реальную пользу. Перед этим его могут переобучить на всех

доступных данных (а не на отдельной обучающей выборке, чтобы проверить на тестовой). Предусматриваются мониторинг качества и поддержка, возможность обновления и дообучения на новых данных.

7. Отчётность, презентация, коммуникация (визуализация данных, модели, результатов)

Это действия, которые сопровождают все этапы: коммуникацию с экспертами, начальством и конечными пользователями алгоритма, написание помощи, отчётов по результатам его работы, протоколирование разработки.

На рис. XX.5 показан один из стандартов выполнения проектов с машинным обучением. Выше мы представили чуть более полное описание, кроме того, на самом деле, возможны переходы с любого этапа на любой. Например, если на втором этапе выяснится, что данные не позволят решить задачу, то задача может быть переформулирована (возвращаемся к первому этапу). Если при оценке качества алгоритма (пятый этап) оно нас не устраивает, то можно собрать новые данные или лучше очистить имеющиеся (возвращаемся на второй этап).

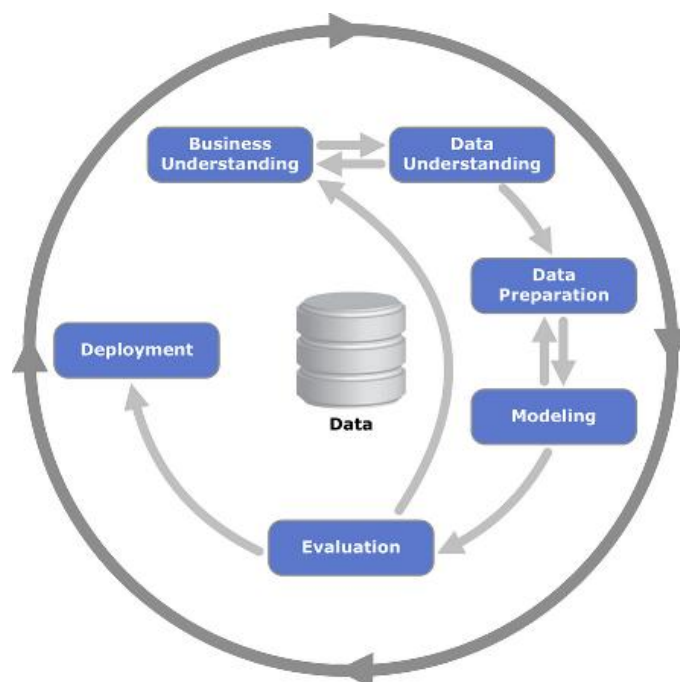


Рис. XX.5. CRISP-DM (Cross-Industry Standard Process for Data Mining¹).

¹ Рисунок взят с https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

Другие виды обучения

Теперь бегло разберём другие виды обучения, многим из них будут посвящены отдельные главы. В **обучении без учителя / с неразмеченными данными / без меток (Unsupervised Learning)** данные (обучающая выборка) выглядят так

$$X_{\text{train}} = \{x_1, \dots, x_m\} \subseteq X,$$

т.е. есть конечный набор объектов без меток, требуется понять «структуру» пространства X . Это может быть по-разному формализовано, например, в виде вопросов:

- Как в пространстве X распределены объекты?
- Можно ли выборку разделить на подвыборки похожих объектов (это т.н. задача кластеризации проиллюстрирована на рис. XX.6)?
- Можно ли эффективно описать объекты/пространство?

и т.д.

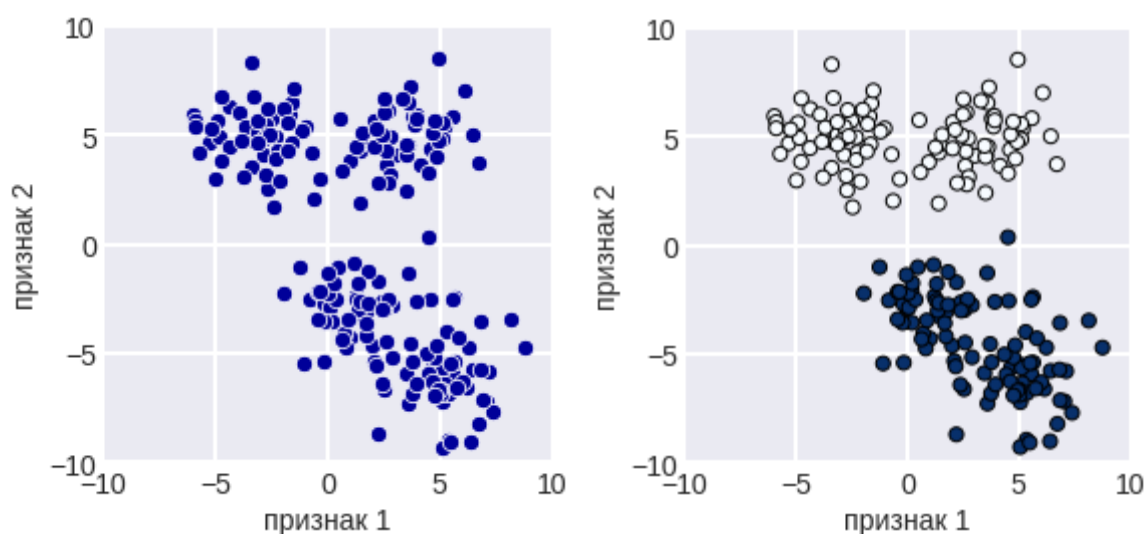


Рис. XX.6. Иллюстрация кластеризации (слева – исходное множество объектов, справа – разные кластеры раскрашены разным цветом).

В **обучении с частично размеченными данными (Semi-Supervised Learning)** обучающая выборка имеет вид

$$X_{\text{train}} = \{(x_1, y_1), \dots, (x_k, y_k), x_{k+1}, \dots, x_m\},$$

т.е. только часть объектов имеет метки (причём обычно небольшая). Как и в задаче с размеченными данными, требуется построить алгоритм, который восстанавливает метку по описанию объекта. На рис. XX.7. показано, почему

бывает выгодно знать неразмеченные данные: есть возможность определить «ту самую структуру пространства» и тогда даже небольшой объём размеченных данных позволяет хорошо решить задачу разметки. Если заранее известна контрольная выборка x'_1, \dots, x'_q (и именно на ней важно качество алгоритма), то это **трандуктивное обучение (transductive learning)**.

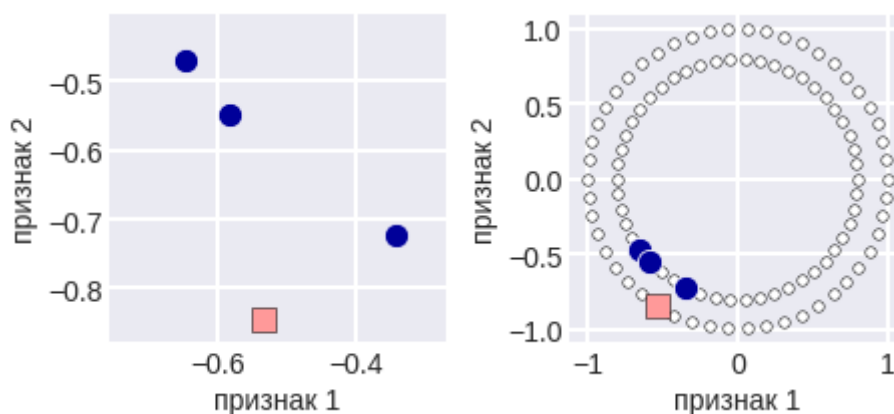


Рис. XX.7. Выборка с метками (слева) и полная выборка с метками и без (справа).

Привилегированное обучение (обучение с привилегированной информацией, Learning Using Privileged Information¹) – обучение с размеченными данными, в котором на обучающей выборки известны также дополнительные описания объектов, например, дополнительные признаки, которые будут неизвестны у объектов тестовой выборки:

$$X_{\text{train}} = \{(x_1, \tilde{x}_1, y_1), \dots, (x_m, \tilde{x}_m, y_m)\},$$

$$X_{\text{test}} = \{x_{m+1}, \dots, x_{m+q}\}.$$

Знание дополнительной информации может помочь лучше понять, как устроена искомая зависимость (значения метки от исходных признаков). Есть также «обучение с частичной разметкой и с привилегированной информацией», см. рис. XX.8

¹ Vapnik V., Vashist A. A new learning paradigm: Learning using privileged information //Neural networks. – 2009. – Т. 22. – №. 5-6. – С. 544-557.

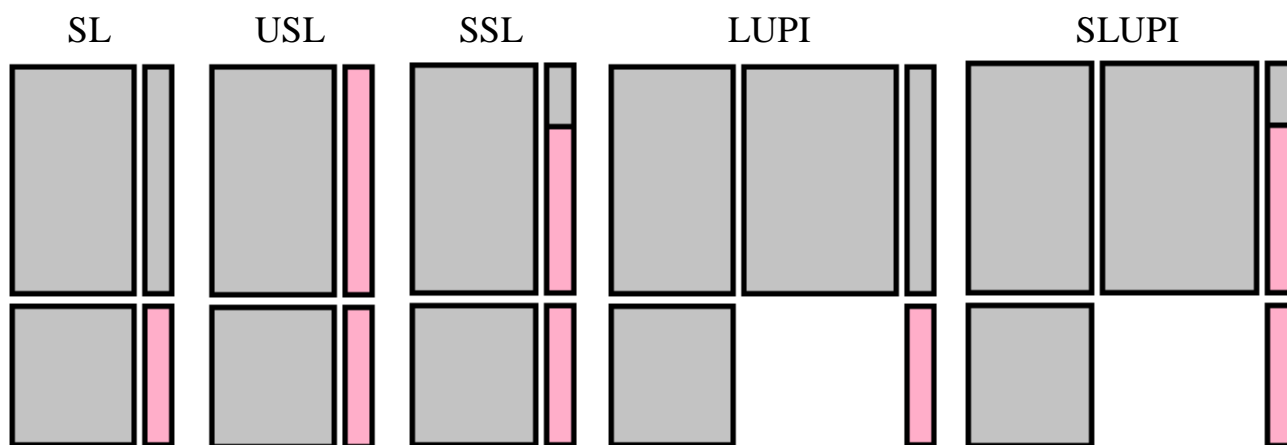


Рис. XX.8. Иллюстрация разных задач, розовым выделены неизвестные значения: SL – обучение с метками, UL – без меток, SSL – с частичной разметкой, LUPI – с привилегированной информацией, SLUPI – с частичной разметкой и с привилегированной информацией.

Обучение с подкреплением (Reinforcement Learning) – обучение агента, который взаимодействует со средой и получает награду за взаимодействие. Необходимо разработать стратегию действий агента, которая максимизирует полученную награду. Пример задачи обучения с подкреплением: научить робота выходить из лабиринта. Робот может совершать некоторые действия, например, шаг вперёд, шаг назад, поворот налево, поворот направо и получать некоторый отклик: «упёрся в стену», «упал в яму», «увидел свет» и т.п. Фактически это и есть метки, но они появляются в результате действий робота и зависят от всей последовательности действий и состояния среды (например, конфигурации лабиринта).

Структурный вывод (Structured output) – обучение с метками, в котором метки это не просто названия классов или вещественные векторы, а сложно устроенные объекты¹, в которых есть связи между элементами (последовательности, графы и т.п.). Примеры задач здесь (с указанием названия и какие целевые зависимости в ней ищутся):

- Грамматический разбор (parsing): текст → дерево,
- Аннотирование изображений (Image Captioning): изображение → текст,
- Транскрипция (Transcription): X → текст,
- Машинный перевод (Machine translation): текст → текст,

¹ Bakır G. (ed.). Predicting structured data. – MIT press, 2007.

- Синтез: выборка → выборка.

Активное обучение (Active Learning) – обучение, в котором мы можем влиять на формирование обучающей выборки, например, есть возможность выбрать объект для разметки (узнать его метку). Необходимость такой постановки вызвана тем, что иногда разметка является очень дорогим (и долгим) процессом, поэтому размеченных данных можно получить немного (мы ограничены бюджетом на разметку).

Онлайн-обучение (Online Learning) – обучение на потоке поступающих данных. Довольно реальная ситуация, в которой обучающая выборка постоянно пополняется. Например, в задаче по поведению пользователя на сайте определить его конечное действие, после визита очередного пользователя становятся известны логи его действий. Фактически при онлайн-обучении модель постоянно дообучается.

Трансферное обучение (Transfer Learning) – обучение, в котором для решения новых задач используются уже готовые решения «старых» (когда есть алгоритм, решающий задачу машинного обучения, и мы пытаемся его использовать для решения похожей задачи). В классическом машинном обучении это встречается редко, зато часто в глубоком (при использовании нейронных сетей).

Мультизадачное обучение (Multitask Learning) – решение одновременно нескольких схожих задач. Из-за схожести задач можно надеяться, что качество решения каждой задачи будет выше, чем при её изолированном решении. Пример мультизадачного обучения: по последовательности кадров видео предсказать следующий кадр, поставить тег (метку) последнему кадру последовательности, оценить качество съёмки по заданной шкале.

Обучение представлений (Representation Learning) – оптимальное представление объектов, в частности, **выучивание признаков (Feature Learning)** – автоматическое получение хороших признаков из сырых данных. Представления (и признаки) могут успешно использоваться для решения других задач машинного обучения¹.

Глубокое обучение (Deep Learning) – решение задач ML с помощью глубоких нейронных сетей.

¹ С этими постановками также связаны обучение многообразий (Manifold Learning), матричные и тензорные разложения (Matrix and Tensor Factorization) и т.п.

Мета-обучение (Meta-Learning) неформально можно определить как «обучение обучаться», например, подбор оптимальных гиперпараметров с помощью машинного обучения. На рис. XX.9 схематично показано, что в классическом обучении с метками мы хотим получить значения параметров (и тем самым обучить наш алгоритм – «чёрный ящик»), а в мета-обучении хотим по обучающей выборке подобрать оптимальный алгоритм для неё – «чёрный ящик» (возможно также и значения его параметров).

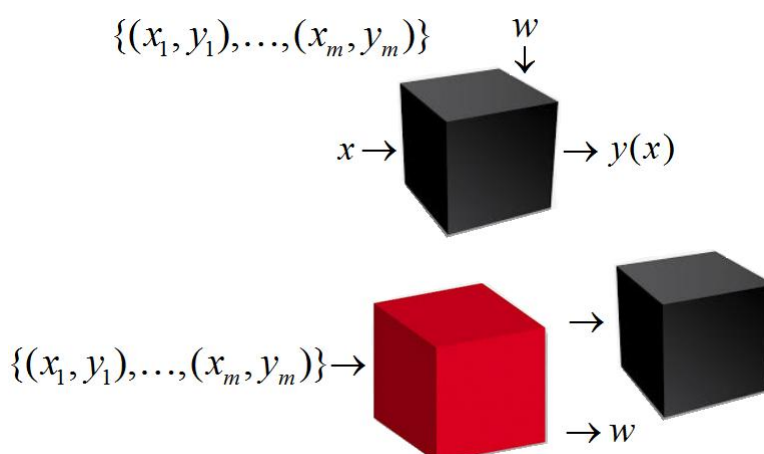


Рис. XX.9. Классическое обучение с учителем (вверху) и мета-обучение (внизу).

В заключение опишем некоторые **сложности в машинном обучении**, с которыми приходится сталкиваться в первую очередь:

- переобучение – это основная теоретическая проблема (мы пока показали один её аспект: обучение не должно превратиться в запоминание обучающей выборки, проблеме будет посвящена **отдельная глава**),
- проблема формализации – надо переформулировать бизнес-проблему в математическую задачу выявления зависимости, выбрать адекватный функционал качества и т.п.
- размеры данных – на практике бывает слишком много или слишком мало объектов или признаков,
- качество данных – низкое качество соответствует невыполнению фундаментальных свойств информации (полнота, корректность, правдивость, ясность и т.п.)
- несоответствие обучения и контроля – это больше, чем проблема репрезентативности выборки, т.к. может быть требование адаптации алгоритма под новые данные (например, в задаче распознавания голоса

мы можем собрать выборку аудиофайлов, в которых проговариваются некоторые предложения, с транскрипциями – предложениями в текстовом виде, затем использовать эту выборку для обучения алгоритма, но алгоритм должен показывать приемлемое качество на аудиозаписях новых спикеров).

Вопросы и задачи

1. В главе перечислены разные виды обучения (learning). Некоторые из них являются постановками задачи (или класса задач), например «обучение с метками». Другие – особенностями решения задач, например «онлайн-обучение» (это обучение с метками, но возникает специфика, связанная с потоком данных). Третьи – приёмами решения других задач, например глубокое обучение – это обучение моделями из конкретного класса (при этом задача может быть как с метками, так и без). Попробуйте разделить разные виды обучения на задачи, спецификации и приёмы.

2. После того, как познакомитесь с основными моделями алгоритмов машинного обучения, попробуйте понять, результаты работы алгоритмов каких моделей показаны на рис. XX.3¹.

3. Попробуйте разобраться, решения каких задач может быть сведено к решению других (возможно, одна задача сводится к набору задач). Например, можно перебрать пары из списка:

- бинарная классификация,
- задача классификации с пересекающимися классами,
- задача классификации с непересекающимися классами,
- задача регрессии.

¹ Правильные ответы (слева-направо и сверху-вниз): 1NN, 3NN, Linear SVM, Kernel SVM, Decision Tree, Deep Decision Tree, Random Forest, Gradient Boosting.

Постановки задач: итоги¹

- Обучение по размеченным данным (с учителем) – восстановление целевой зависимости, заданной прецедентно (обучающей выборкой); качество восстановления формализуется с помощью функции ошибки.
- Объекты в машинном обучении произвольны, но в основном дальше рассматриваем признаковые описания объектов (с помощью вещественных векторов фиксированной размерности).
- Обучение (машинное) – это интеллектуальный подбор параметров, производится минимизацией эмпирического риска (или производной функции) в рамках модели. При этом необходимо обеспечить хорошую обобщающую способность (способность алгоритма хорошо работать на новых данных).
- Есть ключевые отличия машинного обучения от теории оптимизации, важнейшее из которых – контекст (смысл решаемой прикладной задачи).
- Схемы решений задач машинного обучения вполне естественны, алгоритмы и модели довольно просты (например, линейная модель).

Есть много видов машинного обучения, мы начнём с обучения по размеченным данным.

Спасибо за внимание к книге!
Замечания по содержанию, замеченные ошибки
и неточности можно написать в телеграм-чате
<https://t.me/Dyakovsbook>

¹ Можно порекомендовать также классическую книгу Hastie T. et al. The elements of statistical learning: data mining, inference, and prediction. – New York : springer, 2009. – Т. 2. – С. 1-758. Для простого введения в глубокое обучение – Glassner A. Deep learning: A visual approach. – No Starch Press, 2021. Для любителей математики – лекции К.В. Воронцова
[https://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_\(курс_лекций%2C_К.В.Воронцов\).](https://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_(курс_лекций%2C_К.В.Воронцов))